SPEECH COMMAND RECOGNITION ON THE NAO ROBOT

6TH SEMESTER MINI PROJECT REPORT

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

Submitted to: Prof. G.C. Nandi

Submitted by:

(IEC2016012) Nikhil Mundra

(IEC2016026) Agam Dwivedi

(IEC2016027) Bhanu Bhandari

(IEC2016072) Ruchin Agrawal

Problem Statement

In order to make the NAO robot obey simple commands, a novel Convolutional Neural Network based model for the NAO robot has been implemented. In this process, a Hindi speech dataset has also been created. Initially, an LSTM-based model for speech recognition was proposed. It was later found that CNN models for English speech commands gave the best recognition results - and hence were used to trigger a number of actions on the NAO Robot.

Motivation

Speech recognition is defined as the ability of a computer (or a program) to identify parts of the spoken language (words, sub-words, phrases or phonemes) and convert them into a machine-interpretable format. Starting from a rudimentary vocabulary of just numbers, Audrey was invented by Bell Laboratories in 1952.

That was the seminal work which gave rise to the research area of Speech Recognition, and efforts have now taken computers to have superhuman levels of speech recognition, with word error rates (a standard unit of measuring the accuracy of speech recognition) being in the low 5%. [9]

The major breakthrough in the field of Automatic Speech Recognition came in the form of Hidden Markov Models. HMMs are generally used to find the most probable word for a given speech signal. HMMs are robust enough to be used to date in state of the art LVCSR systems.

The introduction of Neural Networks to the field of speech recognition has created a sea change, whether it comes to robustness or accuracy. The application of Neural Networks has allowed an unforeseen depth of analysis to the entire process of speech recognition, thus allowing for computers to come closer to understanding speech as living beings do.

The project intends to explore the use of Recurrent and Convolutional Neural Networks in order to accomplish speech recognition on the NAO robot. The framework of choice after much experimentation is a Convolutional Neural Network model with three convolutional layers and two deep layers.

The learning is done on Mel Frequency Cepstrums of given input audio, and the outputs of the networks are the words which are produced as a part of the learning process.

These speech command words can then be used to trigger a number of actions on the NAO robot, which can allow for remote operation of the robot in a number of circumstances where giving commands to the robot in person is not feasible. Further, it can be integrated with gestures, which gives us the ability to have much more control over the robot.

Related Work

The HMM-DNN model is one of the most widely used models for speech recognition, having superseded the Gaussian Mixture Model-HMM approach since the DNN models are better classifiers than GMMs. Hence, they can generalize much better with a small number of parameters over complex distributions. [10]

LSTMs were first used to model the temporal learning task usually performed using Hidden Markov Models by Graves et al. [8] It was found that for modelling the temporal nature of speech data, which is often sequential in nature (speeches, songs as well as video transcripts are often used as sources for datasets), is modeled in a much better fashion by LSTMs when compared to Hidden Markov Models.

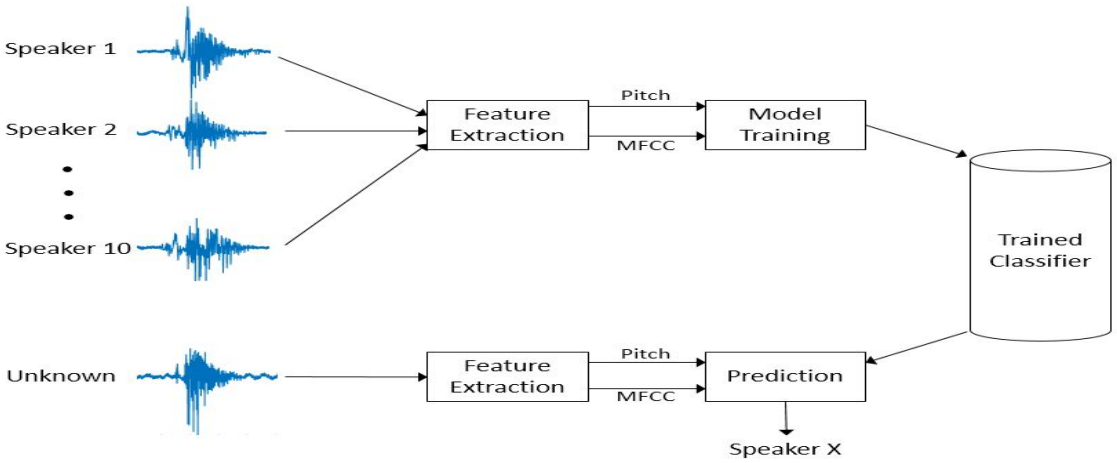
Hence, for state-of-the-art architecture, using LSTMs has become much more beneficial. Particularly considering the case of LSTMs which operate on both past and future data, which gives the LSTM an unprecedented context while learning from the data. This fact is being utilized in a number of attention-based models as well.

However, for detecting command words on the NAO robot, no such previous research work exists. Most speech recognition focuses on being able to recognise large vocabulary, which is infeasible for the NAO as the NAO has limited processing capabilities and such a model would take a long time to recognise speech on the robot. Further, the NAO can only be programmed with a finite number of tasks, which means that recognising command words is a more feasible approach towards performing speech recognition on the NAO.

Terminology

- **MFCC** : The mel filterbank uses 10 linearly spaced triangular filters and then logarithmically spaces them thereafter. The individual bands are weighted for even energy.

  Below is a visualization of a typical mel filterbank. (Figure 1)



(Figure 1: A typical pipeline of using the MFCCs in speech recognition)

- **Deep Neural Network (DNN)**

  A deep neural network is a neural network with a certain level of complexity, a neural network with more than two layers. Each algorithm in the hierarchy applies a nonlinear transformation on its input and uses what it learns to create a statistical model as output. Iterations continue until the output has reached an acceptable level of accuracy. The number of processing layers through which data must pass is what inspired the label deep.

- **Long Short Term Memory (LSTM)**

  Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary

time intervals and the three gates regulate the flow of information into and out of the cell.

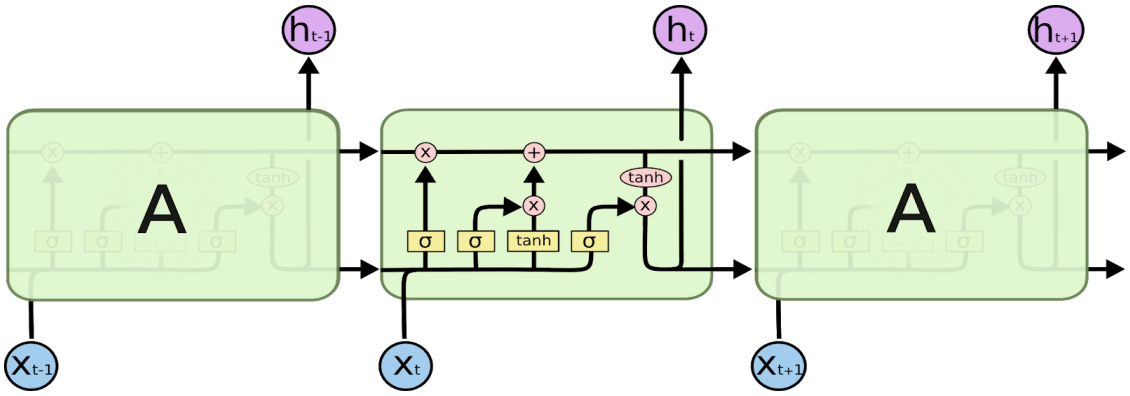


Figure 2. A typical LSTM cell

- **Convolutional Neural Network (CNN)**

  A CNN uses a system much like a multilayer perceptron that has been designed for reduced processing requirements. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers. The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing and natural language processing.

- **Dropout**

  Individual nodes are either dropped out of the net with probability 1-p or kept with probability p, so that a reduced network is left.

- **Batch Normalization**
  - Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). It allows each layer of a network to **learn by itself** a little bit more independently of other layers.
  - We can use **higher learning rates** because batch normalization makes sure that there's no activation that's gone really high or really low.
  - It **reduces overfitting** because it has a slight regularization effect. Similar to dropout, it adds some noise to each hidden layer's activations.

Architecture

State-of-the-art results have been achieved using CNNs, DNNs, and LSTMs, but they are each individually limited in their modeling capabilities. A limitation of using LSTMs alone is that you are relying on them to disentangle the underlying features of variation within the input, which could allow the LSTMs to focus on temporal structure instead.

A good candidate is convolutional neural networks, which are particularly good for disentangling factors. Similarly, DNNs can be used to transform the hidden states to the phoneme state space. We chose to experiment with both CNNs and LSTMs for the given speech data.

Long-Short Term Memory Recurrent Neural Networks (LSTM-RNNs)

In our work, we have firstly worked with Long-Short Term Memory based Recurrent Neural Networks, in an effort to extract the temporal patterns from the waveforms. This gave us a decent accuracy on our testing and validation data, but was not able to perform satisfactorily in real-time testing. We trained the LSTM-RNN model on both Hindi and English datasets after converting raw waveforms to MFCCs.

The proposed Long-Short Term Memory model had the following architecture:

Input Data	MFCCs	Samples	Layers	Dropout	Epochs	Batch Size	Activation
2300 recordings * 28 words	13	100	4, 100 Units each	0.3	50	50	Softmax

For the Hindi language dataset, the following architecture was used:

Input data	MFCCs	Samples	Layers	Drop out	Epochs	Batch Size	Activation
10 recordings per word * 10 speakers * 28 words	13	100 per recording	4, 100 LSTM cells each	0.3	50	50	Softmax

In both the cases, the **Adam** optimiser was used.

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks, in contrast to recurrent neural networks, do not extract the temporal nature of speech. They do, however, focus on extracting more and more higher-order features with each layer, thereby converting the audio file to a two-dimensional frequency-domain image (the MFCC coefficients).

Originally, the proposed model had a number of Dropout layers in the midst of the architecture, which was leading to overfitting and hence did not perform as expected on real-time data. Therefore, instead of Dropout, Batch Normalisation was chosen and the model hence performed as expected relative to the accuracy reported by the model.

The proposed Convolutional Neural Network model is as under:

Mel Frequency Cepstral Coefficients (1700 - 2300 recordings, 1-2 seconds long, for 7 words each)
Two Dimensional Convolutional Layer (16 outputs)
Batch Normalization
Two Dimensional Convolutional Layer (32 outputs)
Batch Normalization
Two Dimensional Convolutional Layer (64 outputs)
Batch Normalization

Max Pooling Layer (Pool size = (2,2))
Flatten 1
Dense 1 () (Dense Layer size = 128)
Dense 2 () (Dense Layer size = 128)
Dense 3 (7 Outputs)

Following the creation and tweaking of these architectures, a **number of functions were created which invoked actions on the robot**. These actions include **walking a prespecified distance, sitting down, standing up, turning left and right, as well as reporting the current status of the robot (battery)**. These were programmed using the **naoqi** library available in Python2.

Datasets

For the project, firstly, we have taken the Tensorflow Speech Commands dataset for the English language. It contains **10 important action words** (“go”, “down”, “up”, “stop”, etc.) as well as 18 auxiliary words (“bed”, “cat”, “one”, “three”, “tree” etc). It has 1700 varied utterances per word including a number of accents, male/female voices, amplitudes, etc.

For the **Hindi language, we made our own dataset** of 10 utterances of 26 words with 10 speakers, making for 2600 iterations. Later on, we introduced a number of augmentations (**nine kinds of augmentations**) to the data, including stretching, changing pitch and speed as well as pitch and speed individually, value augmentations, shifting, and introduction of uniform noise. This brought the total to 2600 iterations.

It was initially intended to train the model to understand continuous speech - however, interacting via continuous speech using the NAO’s inbuilt functionality proved tedious, since there were a considerable number of steps involved in recording audio on the NAO and uploading the model, as well as the problem of incorrect classification leading to unintended behaviour.

Results

Based on the models described above, after further experimentation, the following results have been obtained. The major discovery in the course of the project is that regardless of the model chosen, the real-time accuracy of recognition is lesser as compared to the accuracy obtained on the dataset.

LSTM on English dataset

This performed satisfactorily on the training and testing datasets.

Layers	**Output Shape**	**Parameters**
LSTM	(13,100)	160,400
Dropout	(13,100)	0
LSTM	(13,100)	80,480
Dropout	(13,100)	0
LSTM	(13,100)	80,480
Dropout	(13,100)	0
Dense	7	707

Final Accuracy on 20 Epochs

Loss	**Accuracy**	**Validation Loss**	**Validation Accuracy**
0.0085	82.99%	0.01	79.9%

LSTM on Hindi Dataset

For this particular model and dataset, we obtained low classification accuracy. This model was not trained on the Hindi dataset with as much rigour as it was with the English dataset since it gave worse classification accuracy compared to the English dataset, which gave unsatisfactory results while being tested in real-time.

CNN on Hindi Dataset

This performed well on while the training and testing took place, but it was able to pick up commands properly only intermittently - a problem which can be attributed to the quality of data at hand. Though various methods were used to incorporate variety artificially, it did not give results comparable to the English dataset.

Layers	**Output Shape**	**Parameters**
Conv2d	(19,10,32)	160
Conv2d	(18,9,48)	6192
Conv2d	(17,8,120)	23160
Max pooling	(8,4,120)	0
Dropout 1	(8,4,120)	0
Flatten 1	(3840)	0
Dense 1	(128)	491648
Dropout 2	(128)	0
Dense 2	64	8256
Dropout 3	64	0
Dense 3	26	1690

Final accuracy on 20 epochs:

Loss	**Accuracy**	**Validation Loss**	**Validation Accuracy**
0.3197	89.15%	0.6811	79.94%

```
Epoch 20/20
4306/4306 [==============================] - 5s 1ms/step - loss: 0.3197 - acc: 0.8915 - val_loss: 0.6811 - val_acc: 0.7994
```

CNN on English Dataset

For training the CNN on the English command words, we have used the CNN architecture described above. At the end, we obtained the flow of parameters and intermediate dimensions through each layer as follows:

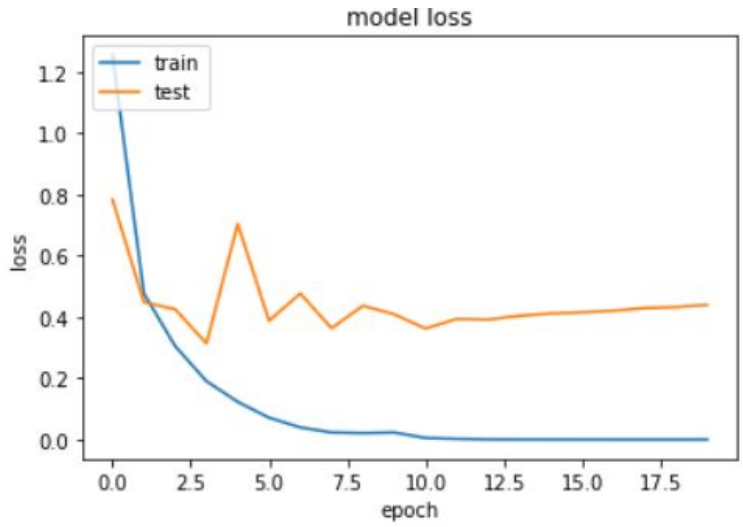
Layer	**Dimensions**	**# of Parameters**
Conv2d	(19,10,32)	160
Batch normalisation	(19,10,32)	128
Conv2d	(18,9,48)	6192
Batch normalisation	(18,9,48)	192
Conv2d	(17,8,120)	23160
Batch normalisation	(17,8,120)	480
Max pooling	(8,4,120)	0
Flatten	(3840)	0
Dense	128	491648
Dense	64	8256
Dense	7	455

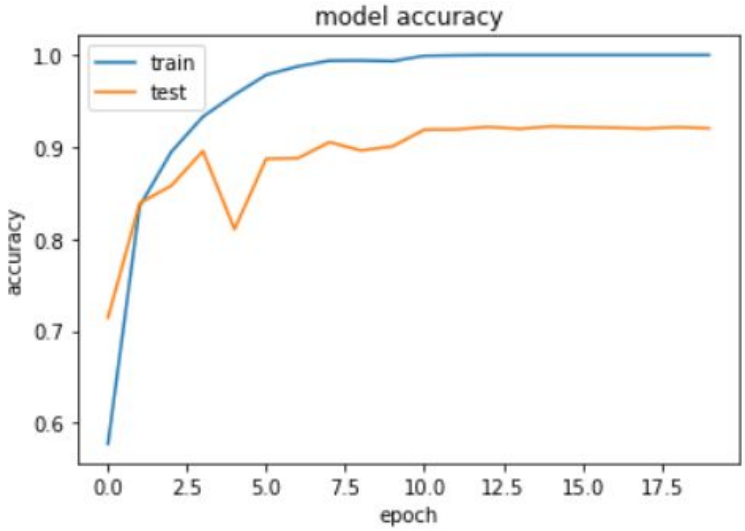
Final accuracy on 10 epochs

Loss	**Accuracy**	**Validation Loss**	**Validation Accuracy**
0.0192	99.34%	0.4890	90.07%

```
Epoch 10/20
13256/13256 [==============================] - 39s 3ms/step - loss: 0.0228 - acc: 0.9934 - val_loss: 0.4090 - val_acc: 0.9007
```

Graphs





The validation accuracy stagnated after 10 epochs whereas validation loss kept decreasing slightly. So we ran our model for 10 epochs to prevent overfitting.

Analysis

Dropout and its implications

A fully connected layer occupies most of the parameters, and hence, **neurons develop co-dependency** amongst each other during training which curbs the individual power of each neuron leading to over-fitting of training data. Thus it is used after a fully connected layer.

The disadvantages of using dropout

- Dropout is generally **less effective** at regularizing convolutional layers.
- What dropout is good at regularizing is becoming **outdated**. It was used when there were models with many fully connected layers as in VGG16.
- Dropout roughly **doubles the number of iterations** required to converge. However, the training time for each epoch is less.

Batch Normalisation in place of Dropout

Dropout can only be applied to the fully-connected region at the end of the convolutional network. For all other regions and layers, dropout should not be used.

- Instead, batch normalization could be used in between the convolutional layers. This will **regularize** the model, as well as make the model more **stable** during training.
- Furthermore, dropout should not be placed between convolutions, as models with dropout tended to perform worse than the control model.

Predicted accuracy vs real-time accuracy

- The accuracy from the data split from the training data is entirely different from what it was tested on. An accuracy on split-test data of 80% gave an accuracy of approximately 50% at the time of real-time testing.
- This could be due to the difference in the distance of the speaker at the time of recording and that during real-time testing.
- Another probable reason is the noise at the time of real-time testing.
- Use of microphone of a headphone, compared to one which was embedded in a laptop, also resulted in different real-time accuracy.

LSTMs vs CNNs: A conceptual comparison

LSTMs are generally used to extract temporal information hidden in the data. This information is effectively lost when the speech command data is converted into mel-frequency coefficients and not further broken down into its constituent units in terms of phonemes. Hence, the LSTM will not be able to learn from the temporal nature of the data.

Further, in Large Vocabulary tasks, certain words are bound to appear one after the other. When commands are considered, there is no such relation between one word and the next, and hence the LSTM won't be as effective as it is when it comes to LVCSR tasks.

CNNs are able to work much better in this regard, because they are able to capture the frequency information inside MFCCs and are able to extract how different words have different frequency components. This is why CNNs work better than LSTMs in our case.

Accuracy with respect to various hyperparameter changes

Kernel Size	Optimizer	Activation Function	Number of Dense Layers	Accuracy
(3,3)	Adam	ReLU	32	0.8919
(3,3)	Adam	ReLU	64	0.8971
(3,3)	Adam	ReLU	128	0.9022
(3,3)	Adagrad	ReLU	32	0.8928
(3,3)	Adagrad	ReLU	64	0.8853
(3,3)	Adagrad	ReLU	128	0.8919
(3,3)	Adam	Softplus	32	0.8999
(3,3)	Adam	Softplus	64	0.8961
(3,3)	Adam	Softplus	128	0.8999
(3,3)	Adagrad	Softplus	32	0.901
(3,3)	Adagrad	Softplus	64	0.8937
(3,3)	Adagrad	Softplus	128	0.8712
(4,4)	Adam	ReLU	32	0.9020
(4,4)	Adam	ReLU	64	0.8946
(4,4)	Adam	ReLU	128	0.8881
(4,4)	Adagrad	ReLU	32	0.8992
(4,4)	Adagrad	ReLU	64	0.8852
(4,4)	Adagrad	ReLU	128	0.8632
(4,4)	Adam	Softplus	32	0.8958
(4,4)	Adam	Softplus	64	0.8968
(4,4)	Adam	Softplus	128	0.9010
(4,4)	Adagrad	Softplus	32	0.8978
(4,4)	Adagrad	Softplus	64	0.8710
(4,4)	Adagrad	Softplus	128	0.8919

Therefore, we can conclude that this model gives consistent results across a number of different hyperparameters. Tuning the dense layer sizes, changing the optimisation functions and kernel sizes as well as the number of units in the dense layer has no considerable impact on the quality of the end result.

Conclusion

In this project, we have accomplished creating a CNN based model on the English language which can directly trigger a number of commands to the NAO robot. This model was the best among the four considered variations over two different languages, and the effects of having a small dataset compared to larger, more diverse dataset were observed. The working of LSTMs and CNNs was also studied thoroughly over the course of the mini project.

Future Milestones

In future, the project can be expanded to cover a larger range of words as well as actions. Further, if provided quality data for speech commands in a number of languages, the model can be used to trigger actions in those languages via transfer learning, taking into account the eccentricities of speech in those languages.

Combining the speech inputs with a variety of other inputs such as touch and gestures can help in developing robust multi-modal systems for command recognition.

References

1. NAOQi documentation: http://doc.aldebaran.com/2-5/index_dev_guide.html
2. Keras documentation: http://keras.io/
3. CLDNN paper: https://research.google.com/pubs/archive/43455.pdf
4. Understanding LSTMs: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Citations

1. G. Pundak, T. N. Sainath, R. Prabhavalkar, A. Kannan and D. Zhao, "Deep Context: End-to-end Contextual Speech Recognition," 2018 IEEE Spoken Language Technology Workshop (SLT), Athens, Greece, 2018, pp. 418-425.
2. T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, “Deep Convolutional Neural Networks for LVCSR,” in Proc. ICASSP, 2013.
3. T. N. Sainath, O. Vinyals, A. Senior and H. Sak, "Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks," 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, QLD, 2015, pp. 4580-4584.
4. M. A. Hossan, S. Memon and M. A. Gregory, "A novel approach for MFCC feature extraction," 2010 4th International Conference on Signal Processing and Communication Systems, Gold Coast, QLD, 2010, pp. 1-5.
5. T. N. Sainath, V. Peddinti, B. Kingsbury, P. Fousek, D. Nahamoo, and B. Ramabhadhran, “Deep Scattering Spectra with Deep Neural Networks for LVCSR Tasks,” in Proc. Interspeech, 2014.
6. T. N. Sainath, B. Kingsbury, A. Mohamed, G. Dahl, G. Saon, H. Soltau, T. Beran, A. Aravkin, and B. Ramabhadran, “Improvements to Deep Convolutional Neural Networks for LVCSR,” in Proc. ASRU, 2013.
7. A. Graves, A. Mohamed and G. Hinton, "Speech recognition with deep recurrent neural networks," *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, 2013, pp. 6645-6649.
8. M. Ravanelli, T. Parcollet, Y. Bengio, "The PyTorch-Kaldi Speech Recognition Toolkit", In Proc. IEEE ICASSP 2019 (Accepted)

Comments